(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 1834904 B
(45)授权公告日 2010.06.09

(21)申请号 200610004500.8

(22)申请日 2006.02.15

(30)优先权数据
11/082,398 2005.03.17 US

(73)专利权人 微软公司
地址 美国华盛顿州

(72)发明人 B·M·诺尔特 M·D·菲尔兹
R·T·沃戴克 S·M·卡洛尔

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 张政权

(51)Int.Cl.
G06F 9/44(2006.01)
G06F 9/46(2006.01)

(56)对比文件
同上.
EP 0762286 B1,2000.02.23, 说明书[0006]-[0013]段.
EP 1391824 A3,2004.09.22, 全文.
EP 1174794 A1,2002.01.23, 说明书第[0005]-[0008],[0010]-[0013]段、说明书附图1.
CN 1175731 A,1998.03.11, 全文.
US 2002/0198925 A1,2002.12.26, 全文.

审查员 胡雅娟

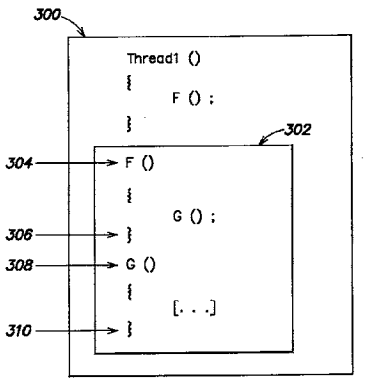
权利要求书 2 页 说明书 12 页 附图 7 页

(54)发明名称

确定处理器在执行一个代码部分中消耗的实际时间量

(57)摘要

本发明提供确定处理器在执行代码部分中消耗的实际时间量的系统和方法。代码部分的实际执行时间可通过将对应于代码部分的上下文切换和/或额外开销时间纳入考虑范围来准确地确定。确定代码部分的实际执行时间可包括记录在代码部分的执行期间所发生的上下文切换和时间值。此信息以及额外开销测量可被用来生成代码部分的实际执行时间,如将在以下更加详细地讨论。例如,可从经过时间减去上下文切换所导致的换出时间间隔和与时间测量相关联的额外开销时间来得到代码部分的实际执行时间。



CN 1834904 B

1. 一种确定多任务操作系统的处理器在执行第一处理线程的一个代码部分中消耗的实际时间量的方法，所述方法包括以下动作：

(A) 接收指示定义所述代码部分的执行开始时间和结束时间的第一时间间隔的第一信息；

(B) 接收指示在所述第一时间间隔内所发生的一个或多个第二时间间隔的第二信息，其中在所述一个或多个第二时间间隔期间所述处理器执行不同于所述第一处理线程的处理线程；

(C) 接收指示在所述第一时间间隔期间获得所述第一信息所消耗的额外开销时间的第三信息；以及

(D) 基于所述第一和第二信息，确定执行所述代码部分中所消耗的实际时间量，包括从所述第一时间间隔减去所述额外开销时间以及所述一个或多个第二时间间隔的总合时间。

2. 如权利要求 1 所述的方法，其特征在于，所述第二信息包括多个第二信息元素，每个所述第二信息元素指定一特定时间、在所述特定时间所述处理器从其切换上下文的旧线程、以及在所述特定时间所述处理器将上下文切换到的新线程，

其中动作 (D) 包括以下动作：(1) 基于所述第二信息元素，为所述一个或多个第二时间间隔的每一个确定所述第二时间间隔的开始时间和结束时间的动作。

3. 如权利要求 1 所述的方法，其特征在于，所述第一信息包括多个第一信息元素，每个所述第一信息元素指定一时间、一处理线程及一事件类型，

其中动作 (D) 包括以下动作：(1) 基于所述多个第一信息元素确定所述第一时间间隔的开始时间和结束时间的动作。

4. 如权利要求 3 所述的方法，其特征在于，所述第二信息包括多个第二信息元素，每个所述第二信息元素指定一特定时间、在所述特定时间所述处理器从其切换上下文的旧线程、以及在所述特定时间所述处理器将上下文切换到的新线程，

其中所述第三信息包括一个或多个第三信息元素，每个所述第三信息元素指定在执行一类事件中所消耗的时间，

其中动作 (D) 还包括以下动作：

(2) 基于所述第二信息元素，为所述一个或多个第二时间间隔中的每一个确定所述第二时间间隔的开始时间和结束时间，

(3) 从所述第一和第二信息元素确定所述处理器在获得所述第一信息中消耗的总的额外开销时间，以及

(4) 基于在动作 (D)(1)-(D)(3) 中所作的确定，确定在执行所述代码部分中所消耗的实际时间量。

5. 如权利要求 1 所述的方法，其特征在于，动作 (A)-(D) 是在计算机系统上执行的。

6. 一种确定多任务操作系统的处理器在执行第一处理线程的一个代码部分中消耗的实际时间量的系统，包括：

用于接收指示定义所述代码部分的执行开始时间和结束时间的第一时间间隔的第一信息的装置；

用于接收指示在所述第一时间间隔内所发生的一个或多个第二时间间隔的第二信息的装置，其中在所述一个或多个第二时间间隔期间所述处理器执行不同于所述第一处理线

程的处理线程；

用于接收指示在所述第一时间间隔期间获得所述第一信息所消耗的额外开销时间的第三信息的装置；以及

用于基于所述第一和第二信息，确定执行所述代码部分中所消耗的实际时间量的装置，进一步包括用于从所述第一时间间隔减去所述额外开销时间以及所述一个或多个第二时间间隔的总合时间的装置。

7. 如权利要求 6 所述的系统，其特征在于，所述第二信息包括多个第二信息元素，每个所述第二信息元素指定一特定时间、在所述特定时间所述处理器从其切换上下文的旧线程、以及在所述特定时间所述处理器将上下文切换到的新线程，

其中用于基于所述第一和第二信息，确定执行所述代码部分中所消耗的实际时间量的装置包括用于基于所述第二信息元素，为所述一个或多个第二时间间隔确定所述第二时间间隔的开始时间和结束时间的装置。

8. 如权利要求 6 所述的系统，其特征在于，所述第一信息包括多个第一信息元素，每个所述第一信息元素指定一时间、一处理线程及一事件类型，

其中用于基于所述第一和第二信息，确定执行所述代码部分中所消耗的实际时间量的装置包括用于基于所述多个第一信息元素确定所述第一时间间隔的开始时间和结束时间的装置。

9. 如权利要求 8 所述的系统，其特征在于，所述第二信息包括多个第二信息元索，每个所述第二信息元素指定一特定时间、在所述特定时间所述处理器从其切换上下文的旧线程、以及在所述特定时间所述处理器将上下文切换到的新线程，

其中所述第三信息包括一个或多个第三信息元素，每个所述第三信息元素指定在执行一类事件中所消耗的时间，

其中用于基于所述第一和第二信息，确定执行所述代码部分中所消耗的实际时间量的装置还包括：

用于基于所述第二信息元素，为所述一个或多个第二时间间隔中的每一个确定所述第二时间间隔的开始时间和结束时间的装置，

用于从所述第一和第二信息元素确定所述处理器在获得所述第一信息中消耗的总的额外开销时间的装置，以及

用于基于所确定的所述第一时间间隔的开始时间和结束时间、所确定的所述第二时间间隔的开始时间和结束时间和所确定的所述处理器在获得所述第一信息中消耗的总的额外开销时间，确定在执行所述代码部分中所消耗的实际时间量的装置。

确定处理器在执行一个代码部分中消耗的实际时间量

背景技术

[0001] 通常，在软件应用程序的开发期间，要对应用程序的一个或多个部分执行性能测试。这一测试常常涉及测量处理器执行软件应用程序的代码的一个或多个部分（例如，函数、过程或其它逻辑组件）所花费（即，消耗）的时间量。例如，可通过记录代码部分执行开始的时间和代码部分执行结束的时间来确定这一时间量。这些时间常常是通过在软件应用程序内的位置（例如，代码部分的开始和结束）包括探针来记录的，执行这些探针导致这些时间值被记录。

[0002] 只基于代码部分执行开始和结束的时间来确定执行时间量不是处理器在执行该代码部分中所消耗的实际时间量的准确表示。记录时间值本身也消耗时间，包括读时间值所需的时间以及将时间值写（即，记录）到诸如易失性存储器或非易失性存储介质等记录介质中所需的时间。获得（包括读和记录）时间值所消耗的时间在本文中称为“额外开销”或“额外开销时间”。

[0003] 例如，开始和结束时间测量可能指示一代码部分消耗了 800 个处理器周期。但是，记录开始时间可能花了三个处理器周期。由此，处理器在执行该代码部分中所消耗的实际时间量（假定没有其它诸如以下将讨论的上下文切换等变数）是 800-3 ＝ 797 个处理器周期。应当认识到，获得结束时间中所消耗的处理器周期不影响所测量的代码部分执行时间的准确性。这是因为结束时间的获得时间是在被记录的结束时间本身之后才发生的。

[0004] 另一个问题在多任务操作系统 (OS) 中出现。多任务 OS 通过交替或交织不同线程的执行，在处理器（例如，中央处理单元 (CPU) 或微处理器）上模拟不同处理线程的并发操作。在一个线程执行了相对很短的一段时间（常被称为“时间片”）以后，OS 中断处理器并将其上下文调整到一不同的线程。将处理器的上下文从一个线程调整或切换到另一个是在本文中被称为“上下文切换”的事件。为代码部分执行的开始和结束时间记录的时间值并没有将其中在开始时间和结束时间之间是否发生了一次或数次上下文切换纳入考虑范围。如果在此时间间隔期间发生了一次或多次上下文切换，则该时间间隔就不是处理器执行该代码部分花费了多少时间的准确表示。即，该时间间隔所反映的时间段将比处理器在执行该代码部分本身中实际消耗的时间段长。

发明内容

[0005] 申请人认识到需要一种用于确定处理器在执行代码部分、特别是当在多任务操作系统上执行时所消耗的实际时间量的可靠系统和方法。

[0006] 由此，本文中描述了确定处理器在执行代码部分中消耗的实际时间量的系统和方法。如在本文中所使用，代码部分的“活动时间”是指处理器在执行该代码部分中所消耗的实际时间量。代码部分的活动时间可通过将对应于该代码部分的上下文切换和 / 或额外开销时间纳入考虑范围来准确地确定。

[0007] 如在本文中所使用，代码部分的“经过时间”是指在该代码部分执行的开始和结束时间之间经过的绝对时间（即，由所记录的该代码的执行开始和结束时间所定义的时间间

隔）。代码部分的经过时间可包括处理器在换出间隔期间所消耗的时间。如本文中所使用，“换出间隔”是指在经过时间间隔期间发生的时间间隔，在该时间间隔期间，处理器执行不同于对应于正被测量的代码部分的处理线程的另一处理线程。经过时间还可包括获得代码部分的开始和结束时间值所导致的额外开销时间。此外，应当理解，代码部分可包括除导致记录开始和结束时间以外的其它探针。例如，可能有其它探针被放置在正被测量的代码部分内的其它代码部分（例如，函数、过程或其它逻辑组件）的开始和结束处。执行这些其它探针产生经过时间内的其它额外开销时间。

[0008] 确定代码部分的活动时间可包括记录在该代码部分的执行期间所发生的上下文切换和时间值。此信息以及额外开销测量可被用来生成代码部分的活动时间，如将在以下更加详细地描述。例如，可从经过时间减去上下文切换所导致的换出时间间隔以及与时间测量相关联的额外开销时间来得到代码部分的活动时间。

[0009] 在本发明的一个实施例中，多任务操作系统的处理器在执行第一处理线程的一个代码部分中所消耗的实际时间量被确定。指示定义该代码部分的执行开始和结束时间的第一时间间隔的第一信息被接收。指示在第一时间间隔内所发生的、期间处理器执行不同于第一处理线程的处理线程的一个或多个第二时间间隔的第二信息被接收。基于第一和第二信息，执行该代码部分中所消耗的实际时间量被确定。

[0010] 在此实施例的一个方面，从第一时间间隔减去了这一个或多个第二时间间隔的总合时间。

[0011] 在此实施例的另一个方面，指示在第一时间间隔期间获得第一信息中所消耗的额外开销时间的第三信息被接收。确定实际时间量包括从第一时间间隔减去该额外开销时间以及这一个或多个第二时间间隔的总合时间。

[0012] 在此实施例的另一个方面，第二信息包括多个信息元素。每个第一信息元素指定一特定时间、在该特定时间处理器从其切换上下文的旧线程、以及在该特定时间处理器将上下文切换到的新线程。确定实际时间量包括基于这些信息元素，为这一个或多个第二时间间隔中的每一个确定该第二时间间隔的开始和结束时间。

[0013] 在此实施例的又一个方面，第一信息包括多个第一信息元素，每个第一信息元素指定一时间、处理线程及事件类型。确定实际时间量包括基于这多个第一信息元素确定第一时间间隔的开始和结束时间的动作。

[0014] 在此实施例的另一个方面，第二信息包括多个第二信息元素。每个第二信息元素指定一特定时间、在该特定时间处理器从其切换上下文的旧线程、以及在该特定时间处理器将上下文切换到的新线程。包括一个或多个第三信息元素的第三信息被接收，其中每个第三信息元素指定在执行一种类型的事件中所消耗的时间。确定实际时间量包括基于这些第二信息元素，为这一个或多个第二时间间隔中的每一个确定该第二时间间隔的开始时间和结束时间。确定实际时间量还包括从第一和第二信息元素确定处理器在获得第一信息中所消耗的总的额外开销时间，以及确定执行该代码部分中所消耗的实际时间量。

[0015] 在另一个方面，在计算机系统上执行上述实施例的一个或多个上述动作和 / 或方面。

[0016] 在本发明的另一个实施例中，提供了一种计算机程序产品。该产品包括计算机可读介质以及存储在该计算机可读介质上的计算机可读信号，这些信号定义了在由计算机执

行时将指示计算机执行在以上段落中所描述的本发明的实施例的方法和 / 或在以上段落中所描述的本发明的一个或多个方面的指令。

[0017] 在本发明的另一个实施例中，提供一种用于确定多任务操作系统的处理器在执行第一处理线程的一个代码部分中所消耗的实际处理器时间量的系统。该系统包括实际时间生成器，用于接收指示定义代码部分执行开始时间和结束时间的第一时间间隔的第一信息，接收指示在第一时间间隔内发生的、期间处理器执行不同于第一处理线程的另一处理线程的一个或多个第二时间间隔的第二信息，以及基于第一和第二信息，生成执行该代码部分中所消耗的实际处理器时间量。

[0018] 在此实施例的一个方面，实际时间生成器用于从第一时间间隔减去这一个或多个第二时间间隔的总合时间。

[0019] 在此实施例的另一个方面，实际时间生成器用于接收指示在第一时间间隔期间获得第一信息中所消耗的额外开销时间的第三信息，以及从第一时间间隔减去该额外开销时间以及这一个或多个第二时间间隔的总合时间。

[0020] 在此实施例的另一个方面，第二信息包括多个信息元素。每个第一信息元素指定一特定时间、在该特定时间处理器从其切换上下文的旧线程、以及在该特定时间处理器将上下文切换到的新线程。实际时间生成器用于基于这些信息元素，为这一个或多个第二时间间隔中的每一个确定该第二时间间隔的开始和结束时间。

[0021] 在此实施例的又一个方面，第一信息包括多个第一信息元素，每个第一信息元素指定一时间、处理线程及事件类型。实际时间生成器用于基于这多个第一信息元素确定第一时间间隔的开始和结束时间。

[0022] 在此实施例的另一个方面，第二信息包括多个第二信息元素。每个第二信息元素指定一特定时间、在该特定时间处理器从其切换上下文的旧线程、以及在该特定时间处理器将上下文切换到的新线程。实际时间生成器用于接收包括一个或多个第三信息元素的第三信息，其中每个第三信息元素指定在执行一种类型的事件中所消耗的时间。实际时间生成器用于基于这些第二信息元素，为这一个或多个第二时间间隔中的每一个确定该第二时间间隔的开始时间和结束时间，从第一和第二信息元素确定处理器在获得第一信息中所消耗的总的额外开销时间，以及基于在动作 (C)(1)-(C)(3) 中所作的确定，来确定执行该代码部分中所消耗的实际时间量。

[0023] 当结合附图考虑以下对本发明、包括对本发明的各方面和实施例的详细描述，本发明的其它优点、新颖特征和目的将会明确，其中附图是示意性的，而并不旨在按比例绘制。在附图中，不同的图中所示出的每个完全相同或几乎相同的组件由单个附图标记表示。为清楚起见，在不必说明即可允许本领域普通技术人员理解本发明的地方，每个图中的每个组件并不是都被标记，本发明的每个实施例或方面的每个组件也不是都被示出。

附图说明

[0024] 图 1 是根据本发明的一些实施例示出一代码部分的执行的经过时间和活动时间的示例的时序图；

[0025] 图 2 是根据本发明的一些实施例示出一种用于确定处理器在执行代码部分期间所消耗的实际时间的方法的示例的流程图。

[0026] 图 3 是根据本发明的一些实施例表示代码的线程的示例的伪码；

[0027] 图 4 是根据本发明的一些实施例示出表示计时器测量的信息元素表的示例的框图；

[0028] 图 5 是根据本发明的一些实施例示出表示上下文切换事件的信息元素表的示例的框图；

[0029] 图 6 是根据本发明的一些实施例示出包括多个表示额外开销测量的信息元素的表的示例的框图；

[0030] 图 7 是根据本发明的一些实施例示出一种基于计时器测量、上下文切换事件和额外开销测量确定处理器在执行代码部分中消耗的实际时间的方法的示例的流程图；

[0031] 图 8 是根据本发明的一些实施例示出一种用于确定处理器在执行一代码部分中所消耗的实际时间的系统的示例的框图；

[0032] 图 9 是根据本发明的一些实施例示出可在其上实现本发明的一些实施例的计算机系统的示例的框图；以及

[0033] 图 10 是根据本发明的一些实施例示出可作为实现本发明的一些实施例的计算机系统的部件使用的存储系统的示例的框图。

具体实施方式

[0034] 本发明的实施例的功能和优点将从下述示例中更加充分地理解。以下示例旨在便于更好地理解本发明的益处及说明本发明的益处，但并不例示本发明的完整范围。

[0035] 如本文中所使用，无论是在书面描述还是在所附权利要求书中，术语“包含”、“包括”、“携带”、“具有”、“含有”、“涉及”等都应被理解为是开放式的，即，意味着包括而非限于。只有过渡语“由……组成”和“本质上由……组成”才应当是封闭式或半封闭式的过渡语，如美国专利局专利审查过程手册（第八版，修订 2，2004 年 5 月）第 2111.03 节中就权利要求书所阐述。

[0036] 示例

[0037] 图 1 是示出代码部分的执行的经过时间和活动时间的示例的时序图 100。图 100 纯粹是一代码部分的时序图的示例性实施例，而并不试图限制本发明的范围。例如图 100 的变体等这样的图的许多其它实现中的任何一种都是可能的，并旨在落入本发明的范围之内。

[0038] 时间轴 112 示出时序图 100 表示从 0 到 12 毫秒 (ms) 的一段时间。应当认识到，可使用例如处理器周期等其它时间单位来示出这一时序图。图形 102 示出处理器在执行线程 1 的时间，从 1.5-3ms，以及从 6-9ms。图形 104 示出从 0.5-1.5ms 及从 9-10ms 获得的指示代码部分的开始和结束执行时间的计时器读。因此，图形 104 表示为该代码部分获得时间值中的额外开销。

[0039] 图形 106 示出期间处理器在执行不同于线程 1 的一个或多个线程的换出时间间隔。

[0040] 基于图形 102、104 和 106，经过时间间隔（图形 108）在 0.5ms 开始并在 9.0ms 结束，这产生了 8.5ms 的经过时间。时序图 100 示出，基于图形 102-108，代码部分的活动时间（图形 110）是 (3-1.5)+(9-6) ＝ 4.5ms。现在将参考图 2-8 描述用于确定图 100 中所示的

代码部分的活动时间的本发明的实施例。

[0041] 图 2 是示出确定处理器在执行代码部分中消耗的实际时间的方法 200 的示例的流程图。方法 200 纯粹是确定代码部分的实际执行时间的方法的一个示例性实施例，而并不试图限制本发明的范围。例如方法 200 的变体等这一方法的许多其它实现的任何一种都是可能的，并且旨在落入本发明的范围之内。

[0042] 在动作 202，为代码部分记录计时器测量。例如，该代码部分所属的线程可在一个或多个位置包括探针，这些位置包括标记该代码部分的开始和结束的位置，还可能包括该代码部分内的位置（例如，其它逻辑组件的开始和结束）。在该代码部分的执行期间，可在每个探针位置处进行计时器测量。这些测量可被记录在记录介质上，诸如本地（例如，易失性）存储器中的临时缓冲区内或诸如磁盘等非易失性存储介质上。表示计时器测量的信息单位的示例在以下参考图 4 描述。

[0043] 暂时离开图 2，图 3 是表示代码的线程 300 的示例的伪码。代码 300 纯粹是代码线程的示例，而并不试图限制本发明的范围。例如线程 300 的变体等代码线程的许多其它的实现中的任何一种都是可能的，并且旨在落入本发明的范围之内。

[0044] 图 3 和即将讨论的图 4-6 在整个说明书中将作为可从其确定代码部分的活动时间的信息的非限制性示例而被参考。应当认识到，提供这些示例是出于说明目的，且它们并不试图限制本发明的范围。其它信息也可被使用。

[0045] 线程 300 包括代码部分 302，它包括函数 F()，其中 F() 包括函数 G()。用于记录函数 F() 的开始和结束（也是代码部分 302 的开始和结束）的探针可被放在代码部分 302 内的位置 304 和 306 处。此外，指示函数 G() 执行开始和结束的探针可被放在位置 308 和 310 处。

[0046] 图 4 是示出表示计时器测量的信息元素表 400 的示例的框图。图 4 纯粹是表示计时器测量的信息元素表的示例性实施例，而并不试图限制本发明的范围。例如表 400 的变体等这样的表的许多其它实现中的任何一种都是可能的，并且旨在落入本发明的范围之内。例如，表 400 可包括多于或少于图 4 中所示的条目和列，并且各条目和列可按不同顺序排列。

[0047] 每个信息元素可包括时间值字段 402、线程 ID 字段 404 和事件字段 406。字段 402（及表 500 的字段 502 和表 600 的字段 604）中所使用的时间单位是处理器（例如，CPU）周期。但是，应当认识到，这些字段中所使用的时间单位可以是诸如毫秒等许多其它类型的单位中的任何一种。例如，信息元素 408 指定在线程 1 中，在 900 个处理器周期处进入函数 F()。执行在以上参考图 3 所描述的代码部分 302 的位置 304 处的探针可产生信息元素 408。应当认识到，尽管在本文中所描述的多个示例中，处理器周期是时间单位，但是也可使用其它时间单位。

[0048] 又如，信息元素 410 指定在线程 2 中，在 1010ms 进入函数 S()。此外，代码部分 302 的位置 308、310 和 306 处的探针分别可产生信息元素 516、520 和 522。

[0049] 回到图 2，在动作 204，上下文切换事件可被记录。即，从一个线程到另一个的处理器上下文的切换可被记录。上下文被切换到的线程或者从其切换上下文的线程中的一个可以是包括该代码部分的线程，而上下文切换到其中或从其切换上下文的一个或多个线程可以是不同于包括该代码部分的线程的另一线程。在一些操作系统上（例如，可从

Microsoft 公司购买的Windows® XP），可提供使上下文切换事件能被捕捉的应用程序编程接口 (API)。

[0050] 如本文中所使用，“应用程序编程接口”或“API”是一个或多个计算机可读指令的指令集，它提供对一个或多个其它定义函数的计算机可读指令的指令集的访问，以使这些函数可被配置成结合应用程序在计算机上被执行。API 可被视为应用程序和特定计算机环境或平台（例如，以下所讨论的任何一个）之间的“胶合物”，并可使程序员能够编出在一个或多个特定计算机平台上或在一个或多个特定计算机环境中运行的应用程序。

[0051] 例如，Microsoft® XP OS包括 Windows 事件跟踪 (ETW)API，它提供用于记录上下文切换事件的特征。此特征可被启用（即，开启）以使其记录上下文切换事件（例如，记录到本地存储器缓冲中或磁盘上）。因此，在 OS 上多个线程的并行执行期间，上下文切换事件、包括在代码部分执行期间的上下文切换事件可被记录。现在将参考图 5 描述表示上下文切换事件的信息元素的示例。

[0052] 暂时离开方法 200，图 5 是示出表示上下文切换事件的信息元素表 500 的示例的框图。表 500 纯粹是表示上下文切换事件的信息元素表的示例性实施例，而并不试图限制本发明的范围。诸如表 500 的变体等这样的表的许多其它实现中的任何一种都是可能的，并且旨在落入本发明的范围之内。例如，表 500 可包括其它条目和 / 或列，并且各条目和列可按不同方式来组织。

[0053] 每个信息元素可在字段 502 中指定一特定时间值，在旧线程 ID 字段 504 中指定从其切换上下文的线程的线程 ID，并在新线程 ID 字段 506 中指定上下文被切换到的线程的线程 ID。例如，信息元素 508 指示，在 1000 个处理器周期（例如，从预定的开始时间起）处，上下文被从线程 1 切换到线程 2，而信息元素 610 指示在 1028 个处理器周期处，上下文被从线程 2 切换到线程 3。

[0054] 在一些实施例中，不是使用字段 504 和 506，而是每个条目可包括单个指定线程 ID 的字段，以及另一指定指示该线程 ID 所标识的线程是被切换到其中（即，换入）还是从其切换（即，换出）的值（例如，标志）的字段。条目的其它实施例也可被使用。

[0055] 回到图 2，在动作 206，特定类型的事件（例如，函数进入、函数退出及其它类型的事件）的额外开销测量可被获得。这些测量可从较早时在例如受控的测试环境中对不同事件类型所作的先前的测量获得。或者，额外开销测量可在代码部分的执行被测量期间被测量和记录。现在将参考图 6 描述表示额外开销测量的信息元素的示例。

[0056] 图 6 是示出包括多个表示额外开销测量的信息元素的表 600 的示例的框图。表 600 纯粹是表示额外开销测量的信息元素表的示例性实施例，而并不试图限制本发明的范围。诸如表 600 的变体等这样的表的许多其它实现中的任何一种都是可能的，并且旨在落入本发明的范围之内。例如，表 600 可包括其它条目和 / 或列，并且各条目和列可按不同方式来组织。

[0057] 表 600 可包括信息元素 606 和 608，其中每个信息元素具有指定事件类型的事件类型字段 602 以及指定与事件类型相关联的额外开销时间的额外开销时间字段 604。例如，信息元素 606 指示函数进入事件类型有 3 个处理器周期的额外开销时间，而信息元素 608 指示函数退出事件类型具有 2 个处理器周期的额外开销。条目 606 和 608 假定每次进入或退出函数时分别消耗相同的时间量（分别是 3 个和 2 个处理器周期）。但是，在一些实施例中，

额外开销可在每次进入或退出函数时被测量和记录，并且此信息可用不同于表 600 中所示的格式来记录。

[0058] 回到图 2，在动作 208，可基于计时器测量、上下文切换事件和额外开销测量，以例如以下参考图 7 所描述的方式来确定代码部分的实际时间。方法 200 或其部分可使用以下参考图 8 所描述的系统 800 来实现。

[0059] 方法 200 可包括其它动作。此外，作为方法 200 的一部分所执行的动作的顺序并不局限于图 2 中所示的顺序，因为这些动作可按其它顺序来执行和 / 或其中一个或多个动作可串行或至少部分地并行执行。例如，可与作为动作 202 的一部分记录一个或多个计时器测量并行地或在其之前记录一个或多个上下文切换事件作为动作 204 的一部分。

[0060] 图 7 是示出例如作为执行方法 200 的动作 208 的一部分，基于计时器测量、上下文切换事件和额外开销测量确定处理器执行代码部分中所消耗的实际时间的方法 700 的示例的流程图。方法 700 纯粹是确定处理器在执行代码部分中消耗的实际时间的方法的一个示例性实施例，而并不试图限制本发明的范围。诸如方法 700 的变体等这样的方法的许多其它实现中的任何一种都是可能的，并且旨在落入本发明的范围之内。方法 700 及其动作可使用以下参考图 8 所描述的系统 800 的一个或多个元件来实现。

[0061] 在动作 702，可从计时器测量确定表示代码部分的经过时间的时间间隔。例如，给定一组计时器测量（例如，在动作 202 所记录的那些测量）和线程标识符，即可确定表示所标识的线程的代码部分的经过时间的时间间隔。

[0062] 例如，使用表 400 中所提供的信息，线程 ID 为 1 的线程表示图 3 中所示的线程 300。动作 702 可包括以下。表 400 中不对应于线程 1 的信息元素可被移除（例如，信息元素 410、412、414 和 418），而其余信息元素可按时间值字段 402 排序。选择已排序的信息元素所表示的最早时间值和最晚时间值来分别定义经过时间间隔的开始时间和结束时间。例如，对表 400 的信息元素应用动作 702 可导致 (900，1550) 的经过时间间隔。动作 702 可由以下参考图 8 所描述的经过时间间隔生成器 830 执行。

[0063] 在动作 704，可从计时器测量和额外开销测量（例如，分别在动作 202 和 206 记录与获得的计时器测量和额外开销测量）确定表示计时器测量的额外开销的一组时间间隔。即，与正被讨论的线程对应的每个计时器测量相关联的额外开销时间可被确定和表示为时间间隔。使用图 3、4 和 6 的示例，动作 704 可包括首先移除表 500 中不对应于线程 1 的信息元素，然后其余信息元素可按时间值字段 502 排序。对于每个对应于线程 1 的信息元素，与事件字段 406 中所标识的事件相关联的额外开销可从表 600 确定。例如，信息元素 508 对应于进入函数 F() 事件，该事件是函数进入事件类型。表 600 的信息元素 606 指示与函数进入事件类型相关联的额外开销是三个处理器周期。额外开销时间间隔随即可被定义如下：（时间值，时间值加上额外开销时间）。因此，对于信息元素 408，可得到额外开销时间间隔 (900，903)。

[0064] 又如，信息元素 520 指定退出函数 G() 事件，该事件具有函数退出的事件类型。表 600 的信息元素 608 指示函数退出事件类型具有两个处理器周期的额外开销。由此，动作 702 可包括从信息元素 420 得到额外开销时间间隔 (1500，1502)。应当认识到，动作 704 可不包括为指定由已排序信息元素（即，对应于正被讨论的线程的信息元素）所指定的时间的最后时间的时间测量信息元素指定额外开销时间间隔。此信息元素可被排除在动作 704

以外是因为与这一测量时间相关联的额外开销时间不会对处理器在执行代码部分中消耗的实际时间添加任何额外开销。即，这一最后时间测量指示经过时间的结束，因此获得这一结束时间测量中的额外开销发生在被记录的结束时间本身以后，因此不会影响活动时间。

[0065] 因此，对表 400 和 600 的信息元素执行动作 704 可导致一组额外开销时间间隔：{(900,903)，(1200,1203)，(1500,1502)}。

[0066] 在动作 706，可从诸如在动作 204 记录的上下文切换事件等上下文切换事件确定一组换出时间间隔。换出时间间隔表示在经过时间期间的时间间隔，期间处理器正在执行不同于该代码部分的线程的另一线程。例如，在图 1 的时序图 100 中，图形 106 指示从 3ms 到 6ms 的换出时间间隔。

[0067] 使用图 5 的表 500 的示例来确定线程 1 的一组换出时间间隔，表 500 中不在字段 604 或 606 中引用线程 1 的信息元素被移除。其余上下文切换信息元素可按时间字段 602 排序。对应于从线程 1 切换到另一线程的每个上下文切换事件（即，“换出事件”）可被标识（例如，由信息元素 608 和 616 表示的上下文切换事件）。对于其中每一个换出事件，切换回线程 1 的对应的上下文切换事件（即，换入事件）可被标识。对于由信息元素 608 和 616 所表示的换出事件，对应的换入事件可被标识为由信息元素 614 和 618 所表示的事件。每个换出事件 / 换入事件对可定义一换出时间间隔。因此，对表 500 中所表示的上下文切换事件执行动作 706 可导致一组换出时间间隔：{(1000,1190)，(1225,1411)}。

[0068] 在动作 708，处理器在执行代码部分中所消耗的实际时间量可从经过时间、额外开销时间间隔集与换出时间间隔集来确定。例如，可从经过时间减去所有额外开销时间间隔和所有换出时间间隔所定义的累计时间来得到代码部分的活动时间。即，活动时间等于经过时间减去（额外开销时间间隔加上换出时间间隔）。

[0069] 使用来自上述动作 702、704 和 706 的示例性结果，线程 300 的代码部分 302 的活动时间：

[0070] ＝ 650-[(900,903)+(1200,1223)+(1500,1502)+(1000,1190)+(1225,1411)]

[0071] ＝ 650-(3+3+2+190+186)

[0072] ＝ 650-384

[0073] ＝ 266。

[0074] 因此，对图 3-6 中所提供的信息应用方法 700 得到 266 个处理器周期的值作为处理器在执行代码部分 302 中消耗的实际时间。

[0075] 方法 700 可包括其它动作。此外，作为方法 700 的一部分所执行的动作的顺序并不局限于图 7 中所示的顺序，因为这些动作可按其它顺序来执行和 / 或其中一个或多个动作可被串行或至少部分地并行执行。例如，动作 702-706 中的任何一个可在另一个以前、以后或与之并行执行。

[0076] 方法 200 和 700 及其动作、以及这些方法和这些动作的各种实施例及变体可由在诸如非易失性记录介质、集成电路存储器元件等一个或多个计算机可读介质或其组合上有形地具体化的计算机可读信号个别或结合地定义。计算机可读介质可以是可由计算机访问的任何可用介质。作为示例，而非限制，计算机可读介质可包括计算机存储介质和通信介质。计算机存储介质包括以用于存储诸如计算机可读指令、数据结构、程序模块或其它数据等信息的任何方法或技术实现的易失性和非易失性、可移动和不可移动介质。计算机存

储介质包括，但不限于，RAM、ROM、EEPROM、闪存或其它存储器技术，CD-ROM、数字多功能盘(DVD)或其它光存储，磁带盒、磁带、磁盘存储或其它磁存储设备，其它类型的易失性和非易失性存储器，任何其它可被用来存储所需信息并可由计算机访问的介质，以及前述的任何适当组合。

[0077] 通信介质通常具体化为诸如载波或其它传输机制等已调制数据信号中的计算机可读指令、数据结构、程序模块或其它数据，并包括任何信息传递介质。术语“已调制数据信号”是指以在信号中将信息编码的方式设置或改变其一个或多个特性的信号。作为示例，而非限制，通信介质包括诸如有线网络或直接连线连接等有线介质，诸如声学、RF、红外和其它无线介质等无线介质，其它类型的通信介质，以及前述的任何适当组合。

[0078] 一个或多个计算机可读介质上所包含的计算机可读信号可定义指令为一个或多个程序的部分，当计算机执行这些指令时，指示计算机执行本文中所描述的一个或多个功能（例如，方法500、方法700或前述的任何动作）和/或其各种实施例、变体和组合。此类指令可用多种编程语言中的任何一种来编写，例如，Java、J#、Graphical Basic、C、C#或C++、Fortran、Pascal、Eiffel、Basic、COBOL、其它编程语言，或上述的各种组合中的任何一种。其上包含此类指令的计算机可读介质可驻留在本文中所描述的任一系统800、900和1000的一个或多个组件上，可被分布在一个或多个此类组件上，并可在其间转移。

[0079] 计算机可读介质可以是便携式的，从而存储在其上的指令可被加载到任何计算机系统资源上以实现本文中所讨论的本发明的诸方面。此外，应当认识到，上述存储在计算机可读介质上的指令并不局限于被具体化为在主计算机上运行的应用程序的一部分的指令。相反，这些指令可被具体化为可被使用以将处理器编程为实现以上所讨论的本发明的诸方面的任何类型的计算机代码（例如，软件或微码）。

[0080] 应当认识到，例如参考图9和10所描述的执行本文中所描述的功能的计算机系统等计算机系统的任何单个组件或多个组件的集合一般可被视为控制这些功能的一个或多个控制器。这一个或多个控制器能用许多方式来实现，诸如用专用硬件和/或固件、使用被编程为使用微码或软件来执行上述功能的处理器，或前述的任何适当组合。

[0081] 图8是示出一种用于确定处理器在执行一个代码部分中消耗的实际时间的系统800的示例的框图。系统800可实现方法200和/或方法700或其部分。系统800纯粹是用于确定处理器在执行代码部分中消耗的实际时间的系统的一个示例性实施例，而并不试图限制本发明的范围。例如系统800的变体等这样的系统的许多其它实现中的任何一种都是可能的，并且旨在落入本发明的范围之内。

[0082] 系统800可包括以下任何元件：处理器816、记录介质818、活动时间生成器828以及其它组件。处理器816可以是能够进行多线程执行的，并可被配置成并行接收一个或多个线程802、806和810。每个线程802、806和810分别可包括一个或多个探针805、808和812。线程802可包括代码部分804，该代码部分804可包括一个或多个探针805，可用来确定执行的活动时间。

[0083] 处理器816可被配置成执行探针805、808和812，并且响应于此，如以上参考图2的动作202所描述地记录时间测量820。此外，处理器816可被配置成生成表示包括任何线程802、806和810在内的多个线程之间的上下文切换的上下文切换事件822。处理器816可响应于处理一个或多个API调用814，如以上参考图2中的动作204所描述地记录上下文

切换事件 822。例如，处理器 816 可由Windows® NT 操作系统控制，它通过使用如上所述的 ETW API 来记录上下文切换事件。记录介质 818 上还可记录额外开销测量 824。

[0084] 活动时间生成器 828 可基于计时器测量 820、上下文切换事件 822 和额外开销测量 824 生成给定线程的活动时间 844。活动时间生成器 828 可包括以下任何组件：经过时间间隔生成器 830、换出时间间隔生成器 834、额外开销时间间隔生成器 838、活动时间引擎 842 以及其它组件。

[0085] 经过时间间隔生成器 830 可被配置成接收计时器测量 820 和线程 ID 826，并如以上参考动作 702 所述地生成经过时间间隔 832。

[0086] 换出时间间隔生成器 834 可被配置成接收线程 ID 826 和上下文切换事件 822，并如以上参考图 7 的动作 704 所述地生成换出时间间隔 836。

[0087] 额外开销时间间隔生成器 838 可被配置成接收线程 ID 826、计时器测量 820 和额外开销测量 824 并如以上参考动作 706 所述地生成额外开销时间间隔 840。

[0088] 活动时间引擎 842 可被配置成接收经过时间间隔 832、换出时间间隔 836 和额外开销时间间隔 840 并生成表示处理器 816 在执行代码部分 804 中消耗的实际时间的活动时间 844。活动时间引擎 842 可被配置成如以上参考动作 708 所述地生成活动时间 844。

[0089] 系统 800 及其组件可使用各种技术中的任何一种来实现，包括软件（例如，C、C#、C++、Java 或其组合）、硬件（例如，一个或多个专用集成电路）、固件（例如，电可编程存储器）或其任意组合。系统 800 的一个或多个组件可驻留在单个设备上（例如，计算机），或者其中一个或多个组件可驻留在若干单独的、离散的设备上。此外，每个组件可被分布在多个设备上，并且其中一个或多个设备可被互相连接。

[0090] 此外，在包括系统 800 的一个或多个组件的一个或多个设备中的每一个设备上，每一个组件可驻留在系统的一个或多个位置。例如，这些系统的组件的不同部分可驻留在设备上的存储器（例如，RAM、ROM、磁盘等）的不同区域。这一个或多个设备中的每一个设备可包括诸如一个或多个处理器、存储器系统、磁盘存储系统、一个或多个网络接口、以及将各组件相互连接的一个或多个总线或其它内部通信链路等多个公知组件等等。系统 100 及其组件可使用诸如以下参考图 9 和 10 所描述的计算机系统等来实现。

[0091] 根据本发明的各实施例可被实现在一个或多个计算机系统上。这些计算机系统可以是诸如基于 Intel PENTIUM 类型处理器、Motorola PowerPC、SunUltraSPARC、Hewlett-Packard PA-RISC 处理器、可从 Advanced Micro Device(AMD) 购买的各种处理器中的任何一种、或任何其它类型的处理器等的通用计算机。应当认识到，一个或多个任意类型的计算机系统可被用来实现本发明的各实施例。

[0092] 根据本发明的一个实施例的通用计算机系统被配置成执行上述一个或多个功能。应当认识到，该系统可执行其它功能，并且本发明并不局限于具有任何一个或一组特定功能。

[0093] 例如，本发明的各个方面可被实现为在诸如图 9 中所示的通用计算机系统 900 中执行的专用软件。计算机系统 900 可包括连接到诸如磁盘驱动器、存储器或其它用于存储数据的设备等一个或多个存储器设备 904 的处理器 903。存储器 904 通常被用来在计算机系统 900 的操作期间存储程序和数据。计算机系统 900 的组件可由互连机制 905 耦合，该机制可包括一个或多个总线（例如，在被集成在同一机器内的组件之间）和 / 或网络（例如，

在驻留在若干单独的离散机器上的组件之间）。互连机制 905 使通信（例如，数据、指令）能在系统 900 的系统组件之间被交换。计算机系统 900 还包括例如键盘、鼠标、跟踪球、话筒、触摸屏等一个或多个输入设备 902，以及例如打印设备、显示屏、扬声器等一个或多个输出设备 901。此外，计算机系统 900 可包含将计算机系统 900 连接到通信网络的一个或多个接口（未示出）（除互连机制 905 以外，或作为其替换方案）。

[0094] 在图 10 中更加详细地示出的存储系统 906 通常包括计算机可读并可写的非易失性记录介质 1001，其中存储了定义要由处理器执行的程序的信号或存储在介质 1001 之上或之中要由程序处理的信息。该介质可以是例如磁盘或闪存。通常，在操作中，处理器使数据从非易失性存储介质 1001 中被读入允许由处理器以比访问介质 1001 的速度更快的速度访问的另一存储器 1002 中。此存储器 1002 通常是诸如动态随机存取存储器 (DRAM) 或静态存储器 (SRAM) 等易失性的随机存取存储器。它可位于存储系统 906 中（如图所示）或可位于系统存储器 904 中（未示出）。处理器 903 一般在集成电路存储器 904、1002 内操纵数据，然后在处理完成以后将数据复制到介质 1001 中。有各种公知技术用于管理介质 1001 和集成电路存储器元件 904、1002 之间的数据移动，并且本发明不限于此。本发明不局限于特定的存储器系统 904 或存储系统 1006。

[0095] 本发明的诸方面可用软件、硬件或固件、或其任意组合来实现。此外，本发明的这些方法、动作、系统、系统元件和组件可被实现为上述计算机系统的一个部分，或可被实现为独立组件。

[0096] 尽管是以例示为可在其上实施本发明的各个方面的一类计算机系统的方式示出计算机系统 900，但是应当认识到，本发明的诸方面不局限于被实现于如图 9 中所示的计算机系统上。本发明的各个方面可在具有不同于图 9 中所示的体系结构或组件的一个或多个计算机上实施。

[0097] 计算机系统 900 可以是能使用高级计算机编程语言编程的通用计算机系统。计算机系统 900 还可使用专门编程的、专用的硬件来实现。在计算机系统 900 中，处理器 903 通常是诸如可从 Intel 公司购买的公知的 Pentium 类处理器等可购买的处理器。还有许多其它处理器可用。此类处理器通常在可以是例如可从 Microsoft 公司购买的Windows® 95、Windows® 98、Windows NT®、Windows® 2000（Windows® ME）或Windows® XP操作系统，可从 Apple Computer 购买的 MACOS 系统 X，可从 Sun Microsystems 购买的 Solaris 操作系统，可从各种来源得到的 Linux 以及可从各种来源得到的 UNIX 等操作系统上执行。各种其它操作系统中的任何一种可被使用。

[0098] 处理器和操作系统一起定义了为其用高级编程语言编写应用程序的计算机平台。应当理解，本发明并不局限于特定计算机系统平台、处理器、操作系统或网络。同样，本领域技术人员应当明确，本发明并不局限于特定编程语言或计算机系统，并且其它适当的编程语言及其它适当的计算机系统也可被使用。

[0099] 该计算机系统的一个或多个部分可被分布在耦合到一通信网络的一个或多个计算机系统（未示出）上。这些计算机系统也可以是通用计算机系统。例如，本发明的各个方面可被分布在配置成向一个或多个客户计算机提供服务的（例如，服务器）或作为分布式系统的一个部分执行总的任务的一个或多个计算机系统中间。例如，本发明的各个方面可在包括分布在执行根据本发明的各个实施例的各种功能的一个或多个服务器系统中间

的组件的客户机－服务器系统上实施。这些组件可以是可执行的、中间的（例如，IL）或解释的（例如，Java）代码，它们使用通信协议（例如，TCP/IP）通过通信网络（例如，因特网）通信。

[0100] 应当认识到，本发明并不局限于在任何一个或一组特定系统上执行，并且本发明并不局限于任何特定的分布式体系结构、网络、或通信协议。

[0101] 本发明的各个实施例可使用诸如 SmallTalk、Java、J#(J-Sharp)、C ++、Ada 或 C#(C-Sharp) 等面向对象的编程语言来编程。其它面向对象的编程语言也可被使用。或者可使用功能的、脚本的和／或逻辑编程语言。本发明的各个方面可在非编程环境（例如，以 HTML、XML 或其它格式创建的文档，当在浏览器程序的窗口中查看这些文档时，它们呈现图形用户界面 (GUI) 的方面或执行其它功能）中实现。本发明的各个方面可被实现为编程的或非编程的元素，或其任何适当的组合。此外，本发明的各个实施例可使用可从 Microsoft 公司获得的 Microsoft.NET 技术实现。

[0102] 在已描述了本发明的一些示例性实施例以后，本领域技术人员应当明确，前述纯粹是示例性而不是限制性的，它们仅是作为示例而给出的。许多修改和其它示例性实施例是在本领域普通技术人员范围之内的，并且构想它们落入本发明的范围之内。特别地，尽管本文中所给出的许多示例涉及方法动作或系统元件的特定组合，但是应当理解，那些动作和那些元件能以其它方式组合来实现相同的目的。仅结合一个实施例讨论的动作、元件和特征并不旨在被排除在其它实施例中的类似角色之外。此外，对于所附权利要求书中所述的一个或多个手段加功能限制，这些手段并不旨在被局限于本文中所解释的用于执行所述功能的手段，而是旨在将任何已知或有待开发的、用于执行所述功能的等效手段覆盖在范围之内。

[0103] 在所附权利要求书中使用序数项“第一”、“第二”、“第三”等等来修饰权项元素本身并不意味着任何优先级、优先权、或是一个权项元素排在另一个之前、或者执行方法的动作的时间顺序，而是纯粹被用作标记以将具有特定名称的一个权项元素与另一具有相同名称（除了使用了序数项以外）的元素区别开来，以区别各权项元素。

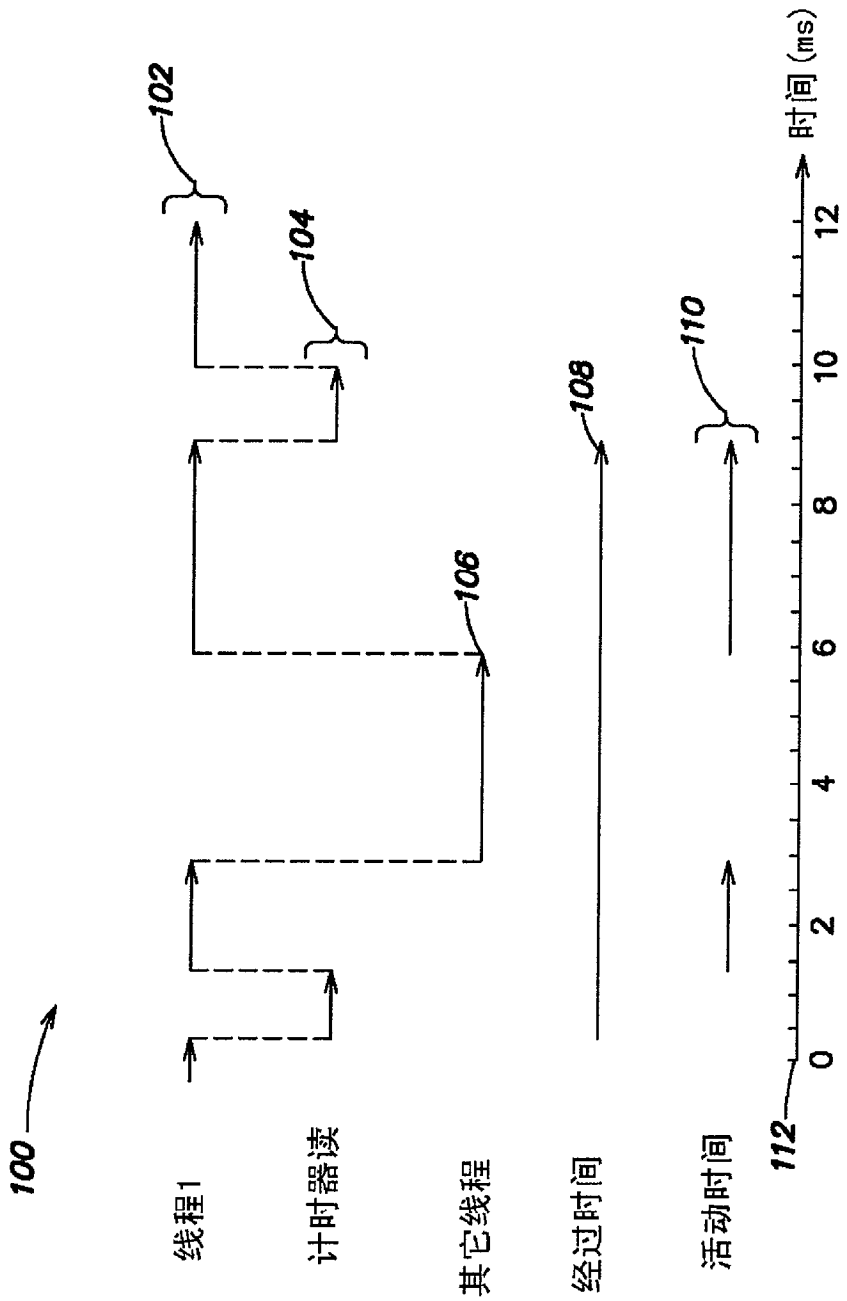

100
线程1
计时器读
其它线程
经过时间
活动时间
102
104
106
108
110
112
0
2
4
6
8
10
12
时间(ms)

图 1

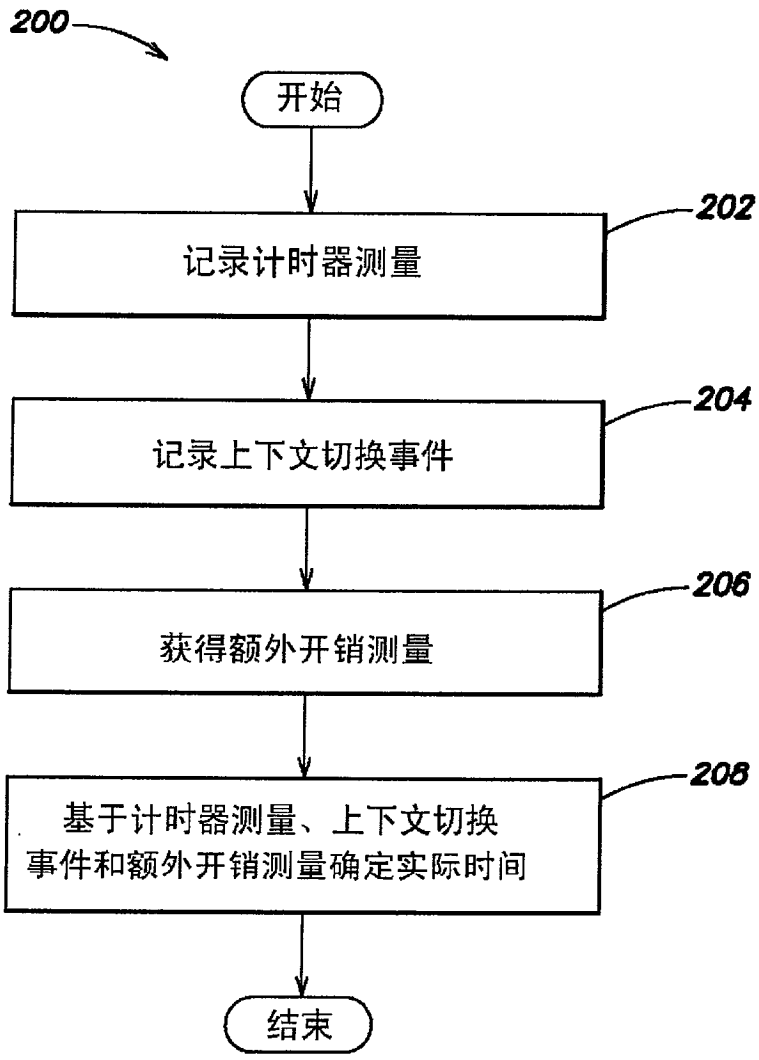

200
开始
202
记录计时器测量
204
记录上下文切换事件
206
获得额外开销测量
208
基于计时器测量、上下文切换事件和额外开销测量确定实际时间
结束

图 2

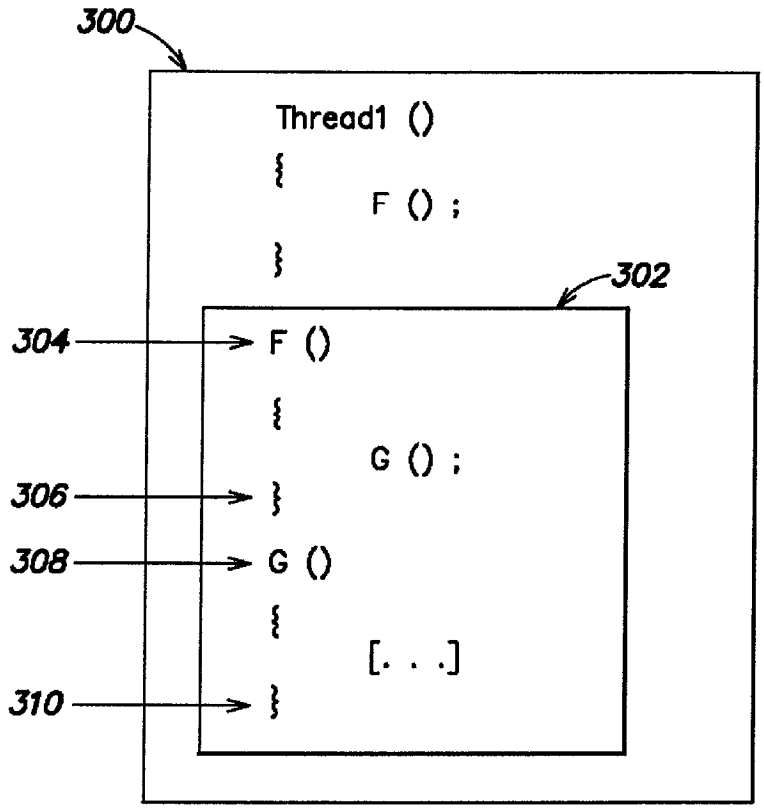
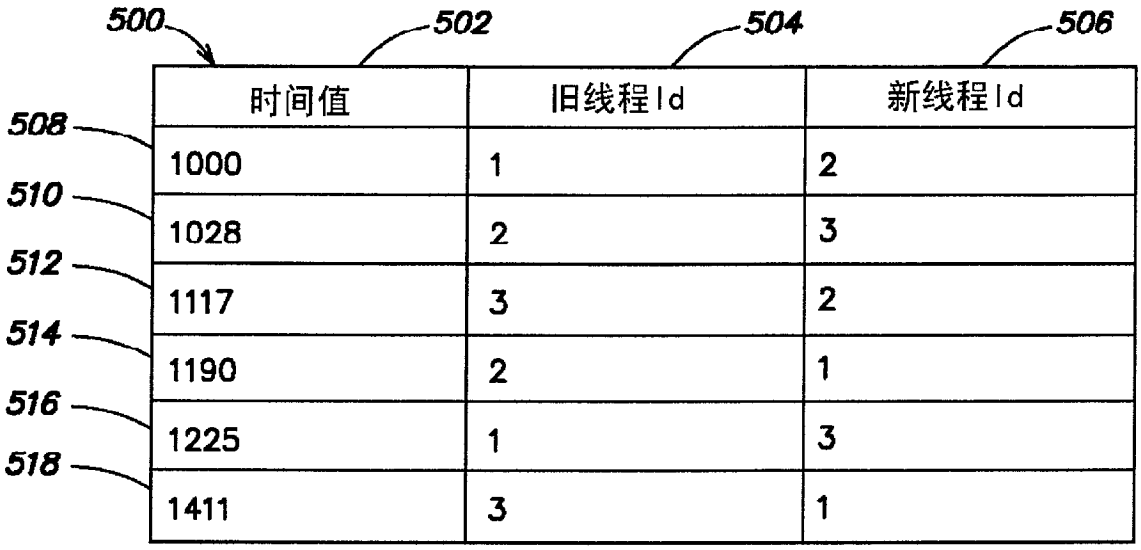


图 3

400 402 404 406

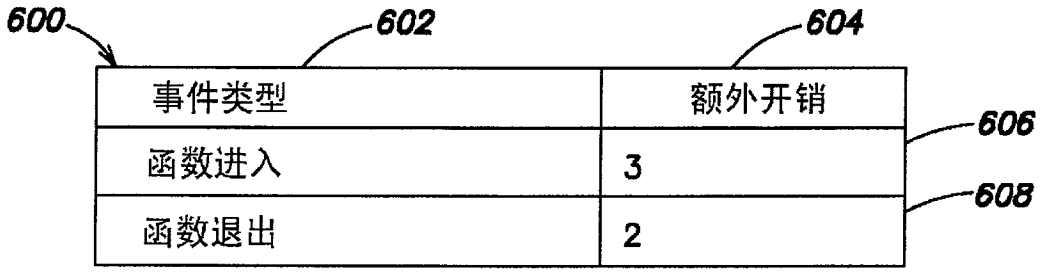
	时间值	线程Id	事件
408	900	1	进入函数 F()
410	1010	2	进入函数 S()
412	1030	3	进入函数 T()
414	1120	2	退出函数 S()
416	1200	1	进入函数 G()
418	1300	3	退出函数 T()
420	1500	1	退出函数 G()
422	1550	1	退出函数 F()

图 4



时间值	旧线程Id	新线程Id
1000	1	2
1028	2	3
1117	3	2
1190	2	1
1225	1	3
1411	3	1

图 5



事件类型	额外开销
函数进入	3
函数退出	2

图 6

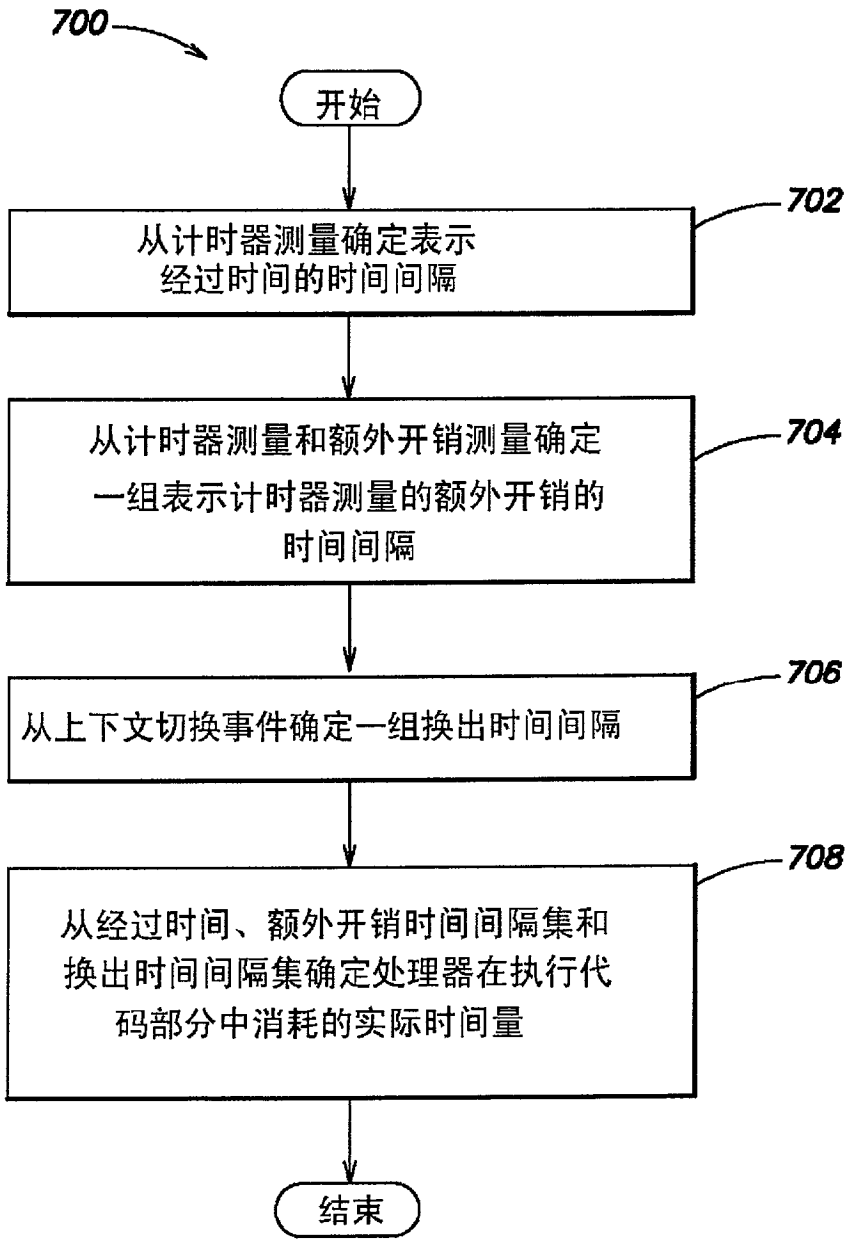

700
开始
从计时器测量确定表示经过时间的时间间隔
702
从计时器测量和额外开销测量确定一组表示计时器测量的额外开销的时间间隔
704
从上下文切换事件确定一组换出时间间隔
706
从经过时间、额外开销时间间隔集和换出时间间隔集确定处理器在执行代码部分中消耗的实际时间量
708
结束

图 7

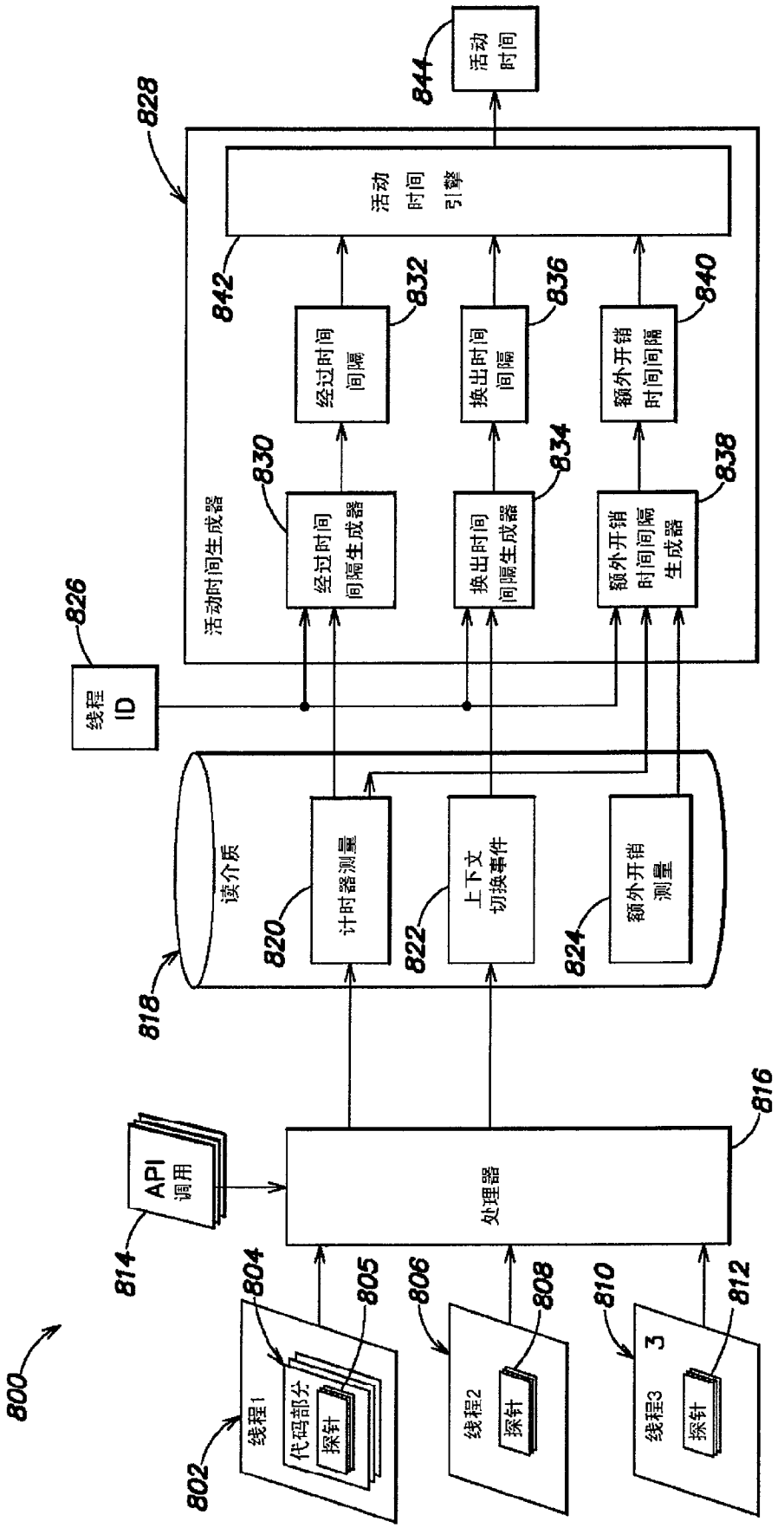

800
802
线程1
代码部分
探针
804
805
806
线程2
探针
808
810
线程3
探针
812
814
API
调用
816
处理器
818
读介质
820
计时器测量
822
上下文
切换事件
824
额外开销
测量
826
线程
ID
828
活动时间生成器
830
经过时间
间隔生成器
832
经过时间
间隔
834
换出时间
间隔生成器
836
换出时间
间隔
838
额外开销
时间间隔
生成器
840
额外开销
时间间隔
842
活动
时间
引擎
844
活动
时间

图 8

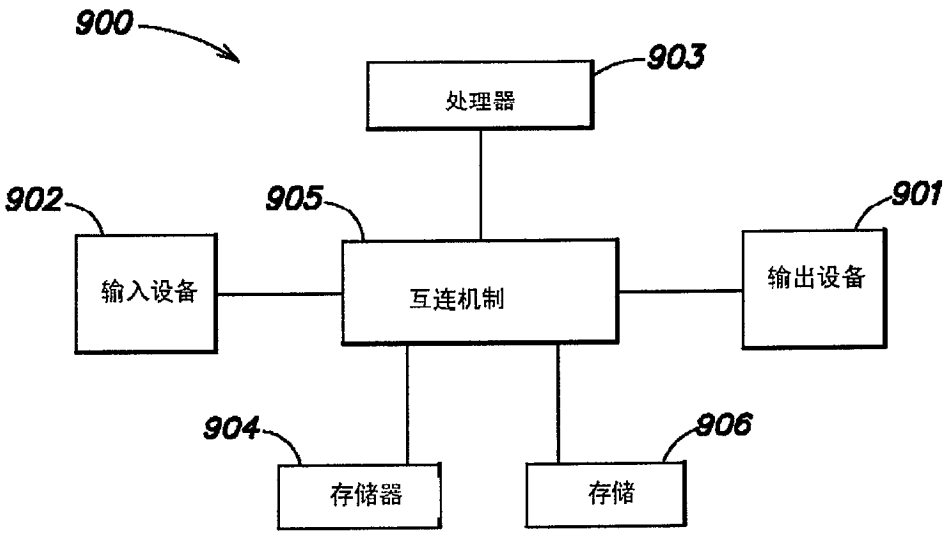

900
903
处理器
902
905
901
输入设备
互连机制
输出设备
904
906
存储器
存储

图 9

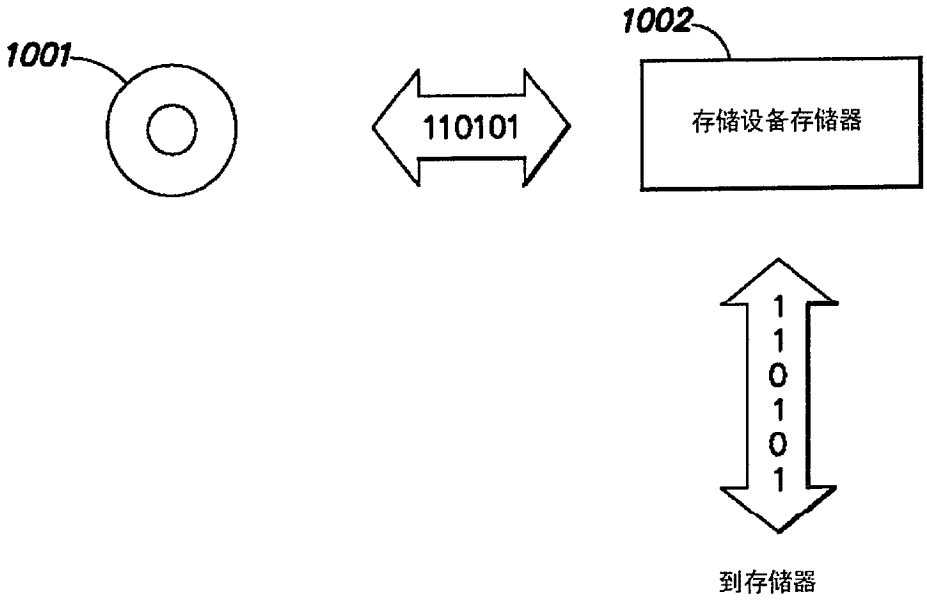

1001
110101
1002
存储设备存储器
1
1
0
1
0
1
到存储器

图 10